OULUN AMMATTIKORKEAKOULU

Henri Junttila

UNITY3D:N KÄYTTÖ PELIKEHITYKSESSÄ

Unity3D:n käyttö pelikehityksessä

Henri Junttila
Opinnäytetyö
Kevät 2015
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä(t): Henri Junttila
Opinnäytetyön nimi: Unity3D:n käyttö pelikehityksessä
Työn ohjaaja(t): Pekka Alaluukas
Työn valmistumislukukausi ja -vuosi: Kevät 2015 Sivumäärä: 40

Alun perin opinnäytetyön tavoitteena oli vain tutustuttaa käyttäjä Unity3D:n kehitysympäristöön, mutta työn edetessä siihen lisättiin myös esimerkkiprojekti. Opinnäytetyössä esitellään Unity3D:n lisäksi työkaluja, jotka toimivat sen kanssa hyvin yhteen.

Esimerkkiprojektissa kokeiltiin Terrain-komponentin käyttöä maastona peliprototyypissä, jossa vaaditaan maaston tosiaikaista muokkautumista. Työn esimerkkiprojektia varten tehtiin grafiikat itse käyttäen 3D-mallinnustyökalu Blenderiä 3D-mallinnukseen ja kuvankäsittelyohjelma Photoshopia pintakuvioiden ja muiden grafiikoiden tekemistä varten. Esimerkkiprojektin ympäristönä toimi vedessä kelluva jäälohkare. Vesi toteutettiin kolmannen osapuolen komponentilla.

Työssä esiteltiin Unity3D:n perusominaisuudet ja lisäksi tehtiin esimerkkiprojekti, jossa tutustuttiin Terrain-komponenttiin ja kokeiltiin sitä. Komponentti toimi peliprototyypissä, mutta osoittautui rajoittuneeksi jatkokehitystä ajatellen. Esimerkkiprojektia tehdessä todettiin Unity Editorin lyhentävän kehitysaikaa.

Asiasanat: Unity3D, pelikehitys, 3D-mallinnus, pintakuviointi

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Development

Author(s): Henri Junttila
Title of thesis: Using Unity3D in game development
Supervisor(s): Pekka Alaluukas
Term and year when the thesis was submitted: Spring 2015 Pages: 40

Originally the objective of this thesis was to create an introduction to Unity3D development. As the work progressed an example project was added. The thesis also includes introductions to a few tools that integrate seamlessly with Unity3D.

The example project concentrated to Unity3D's Terrain-component. The main objective was to test the component's suitability in a game prototype that requires real-time terrain deformation. All the graphics were made specifically for the project using 3D graphics editor Blender as a 3D-modelling software and raster graphics editor Photoshop for texturing and other graphics. Using a 3rd-party package as part of the project was tested for water effects.

In the thesis basic features of Unity3D got covered and an example project with Unity3D's Terrain-component was created and tested successfully. The component worked out for the game prototype but was deemed limited for future development. During the example project, Unity3D and specifically its editor's positive effect on development time when prototyping was noted significant.

Keywords: Unity3D, game development, 3D-modelling, texturing

ALKUSANAT

Tämän opinnäytetyön tekemiseen on kulunut huomattava määrä aikaa, pääosin tästä työstä riippumattomista asioista johtuen. Opinnäytetyön aikana Unity3D on kehittynyt 3.5-versiosta 4.6-versioon, joka on toiminut tärkeänä syynä työn rajaukselle pelikehityksen kannalta vain tärkeimpiin osa-alueisiin. Työnohjaajan kärsivällisyys on ollut kiitoksen arvoinen.

9.3.2015

Henri Junttila

SISÄLLYS

TERMIEN JA KÄSITTEIDEN SELITYKSET

App Store	Applen sovelluskauppa
Apple	Monikansallinen tietotekniikkaan erikoistunut yhtiö
Basic	Vuonna 1964 kehitetty yksinkertainen ohjelmointikieli
Binääri	Yksinkertainen lukujärjestelmä
Boo	Oliopohjainen ohjelmointikieli
Build	Ken Silvermanin kehittämä pelimoottori
C/C++	Yleisimmät ohjelmointikielet
DrawCall	Näytönohjaimen piirtokutsu
Duke Nukem 3D	Vuonna 1996 julkaistu ampumispeli
float	Tietotyyppi liukuluvuille
iOS	Applen mobiilikäyttöjärjestelmä
Java	Oliopohjainen ohjelmointikieli
Mac	Applen kehittämä Macintosh-tietokone
MOD-musiikki	80-luvulla suosiossa ollut tietokonemusiikki
MPEG	Median pakkausmuoto
Nvidia	Monikansallinen tietotekniikka yhtiö, joka on erikoistunut näytönohjaimiin
Ogg Vorbis	Musiikin pakkausmuoto
Pascal	60-luvulla kehitetty proseduraalinen ohjelmointikieli
PC	Henkilökohtainen tietokone

SI-järjestelmä	Kansainvälinen mittayksikköjärjestelmä, joka perustuu metrijärjestelmään
Windows	Microsoftin kehittämä käyttöjärjestelmä
WYSIWYG	Tulee sanoista “What you see is what you get” ja tarkoittaa, että editorissa tehty vastaa lopputulosta

1 JOHDANTO

Olen ollut koukussa videopeleihin siitä asti, kun vanhempani ostivat Nokian MikroMikko-tietokoneen yhdeksänkymmentäluvun alkupuolella. Pitkään pelimaailmani rajoittui kaupallisiin peleihin ja nimenomaan tietokonealustalle, mutta sittemmin olen tutustunut myös itsenäisesti tuotettuihin pienempiin tietokonepeleihin sekä muiden alustojen peleihin. Tietokonepelien sielunelämään tutustuin ensimmäiseksi vuonna 1996 julkaistun Duke Nukem 3D -hittipelin mukana tulleen silmiä avaavan Build-karttaeditorin myötä.

Buildin jälkeen tutkin pelaamieni pelien muokkausta enemmänkin, ja ennen pitkään halu luoda jotain kokonaan omaa kasvoi liian suureksi. Ensimmäiseksi yritin harjoitella ohjelmoimaan C-kielellä, mutta 14.4k-modeemi yhdistettynä 12-vuotiaan englannin kielen ja matematiikan oppeihin tarkoitti, ettei ruudulle ilmestynyt juuri tekstiä enempää.

Sitten tutustuin pelien tekemiseen suunniteltuihin ohjelmistoihin. Tällaisen ohjelmiston ominaispiirre on, ettei matalan tason ohjelmointia tarvita ollenkaan, vaan kehittäjä voi sen sijaan keskittyä pelimekaniikkaan ja sisältöön. Ohjelmistoja löytyi jo tuolloin laidasta laitaan. Monipuolisimmat tarjosivat kääntäjän, kun taas yksinkertaisimmat olivat puhtaita WYSIWYG-editoreita. Useimmat kääntäjällä varustetut perustuivat Basic-kieleen tai olivat kovasti samankaltaisia, kuten esimerkiksi DarkBasic ja BlitzBasic.

Myöhäisteini-ikäisenä ajauduin muihin aktiviteetteihin. Osaksi näin kävi, koska DOS-käyttöjärjestelmä vei valtavirrasta kuollessaan mukanaan suuren osan työkaluista, ja osaksi, koska kuvatarkkuuden nousu ja 3D-vallankumous tekivät grafiikan luonnista huomattavasti enemmän taitoa ja ennen kaikkea aikaa vievää.

Pelien ja Unity3D:n pariin ajauduin takaisin opiskeluihin kuuluvan harjoittelupaikkani kautta. Huomasin Unity3D:n yhdistävän nuoruudessa tapaamieni kehitysympäristöjen monipuolisuuden ja helpon lähestyttävyyden kaupallisten pelien tarpeita vastaavaan pelimoottoriin.

Tämä opinnäytetyö perustuu suurelta osin omaan oppimisprosessiini ja kokemuksiini. Tarkoituksena on esitellä Unity3D vaihtoehtona moottoria ja kehitysympäristöä valittaessa, käydä läpi yleisiä harha-askelia ja luoda esimerkkiprojektissa peliprototyyppi. Peliprototyyppi keskittyy reaaliaikaisen maaston muokkauksen kokeiluun käyttäen Unity3D:n omaa Terrain-komponenttia.

2 UNITY3D

Unity3D:tä kehittää monikansallinen Unity Technologies, jonka pääkonttori sijaitsee Yhdysvalloissa. Unity3D sai alkunsa vuonna 2001, kun kolme tanskalaista pelinkehittäjää huomasivat, kuinka paljon aikaa ja resursseja kaupalliseksi kelpaavan pelimoottorin ja sitä tukevien työkalujen tekoon kului. Saman ongelma kanssa etenkin pienet resurssit omistavat pelistudiot kamppailivat. (1; 2.)

Unity3D:n ensimmäinen versio julkaistiin kaikessa hiljaisuudessaan vuonna 2005. Se oli ominaisuuksiltaan vaatimaton verrattuna nykyiseen ja tuki vain Applen Mac-alustaa. Mac ei ollut merkittävä pelialusta, mutta versio suositummalle Windows-alustalle ja selaimille julkaistiin nopeasti perään. Applen julkaistessa ohjelmistokauppa App Storen vuonna 2008 Unity Technologies kehitti nopeasti tuen Applen iPhone-puhelimelle. Tuen myötä Unity3D:n suosio kasvoi ja siitä muodostui merkittävä pelimoottori. (1; 2.)

Vuosien mittaan pelimoottorista on kehittynyt hienostuneempi. Tänä päivänä siitä löytyy sisäänrakennettuna yleisimmät tarvittavat ominaisuudet, kuten animaatio-, fysiikka- ja hiukkasmoottorit. Fysiikkamoottori perustuu Nvidian omistamaan PhysX-moottoriin. Vuonna 2011 Unity Technologies osti animaatioihin erikoistuneen Mecanim-yrityksen parantaakseen animaatioiden tasoa. Yritysoston jälkeen Unity3D:n animaatiomoottorin nimi muuttui Mecanimiksi. (1; 2.)

Tällä hetkellä Unity3D:n tukemiin alustoihin kuuluvat iOS, Android, Windows, BlackBerry 10, OS X, Linux, selaimet, Flash, PlayStation 3, PlayStation Vita, Xbox 360, Windows Phone 8, Wii U, PlayStation 4 ja Xbox One (1).

2.1 Ilmais- ja ammattilaislisenssin erot

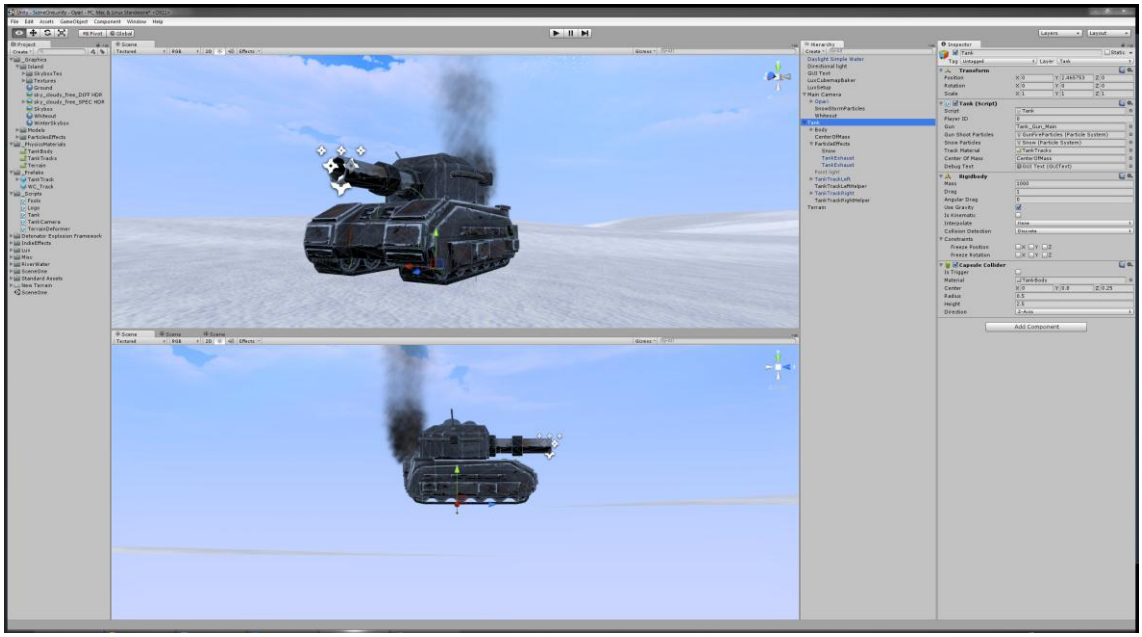
Unity3D:n ilmaisen ja ammattilaislisenssin välillä on useita pieniä eroja (3). Pääosin ammattilaislisenssi tuo kehittyneempiä ominaisuuksia, jotka eivät ole pakollisia mutta parantavat pelin kilpailukykyä. Kehitysympäristön kannalta tärkeitä eroja ovat Team License -lisäosa, jolla saa sisäisen versionhallinnan, ja Profiler, jolla voi monitoroida eri käskyjen resurssien käyttöä (4). Ilmaisen ja ammattilaislisenssin erot vaihtelevat uusien versioiden mukana, ja esimerkiksi

versionhallintaa varten voi tätä nykyä käyttää myös ulkoista versionhallintaa. Tämän lisäksi ilmaisversiolla tehdyissä peleissä esiintyy Unity3D-logo. Logo ilmestyy joko peliä aloittaessa tai on koko ajan ruudun kulmassa. Android- ja iOS-käyttöjärjestelmille on lisäksi omat ammattilaislisenssit, jotka ovat lisäosia normaaliin ammattilaislisenssiin.

Jokaisen ammattilaislisenssin voi ostaa joko 1 140 euron kertamaksulla tai 57 euron kuukausimaksulla. Kuukausimaksu ei ole osamaksua kertamaksusta, vaan kuukausikohtainen lisenssi. Niin ilmaisella kuin ammattilaislisenssillä tehtyjä pelejä voi myydä ilman rojalteja, mutta jos yrityksen liikevaihto ylittää 100 000 Yhdysvaltain dollarin rajan, on ammattilaislisenssin hankinta pakollista. (5.)

2.2 Unity Editor

Unity3D:n editori on nimetty yksinkertaisesti Unity Editoriksi (kuva 1). Se koostuu muokattavista ikkunoista ja niiden välilehdistä. Välilehdissä voi olla esimerkiksi objektien muokkaukseen tarkoitettu Inspector-työkalu, objektilistauksia tai 3D-näkymiä. (6.)



KUVA 1. Unity3D:n editorin graafinen käyttöliittymä

Ensinäkymältään editori vaikuttaa samalta kuin minkä tahansa pelin kenttäeditori, eikä käyttöliittymän vahva vedä ja pudota -periaate anna kuvaa monipuolisesta työkalusta. Todellisuudessa yksinkertaisen pinnan alta kuitenkin löytyvät monipuoliset työkalut pelikehitykseen. Eräs Unity3D:n vahvuuksista on sen editorin yksinkertaisuuden tuoma helppo lähestyttävyys, ja samoin se, että kaiken voi vaihtoehtoisesti tehdä ohjelmakoodinkin kautta.

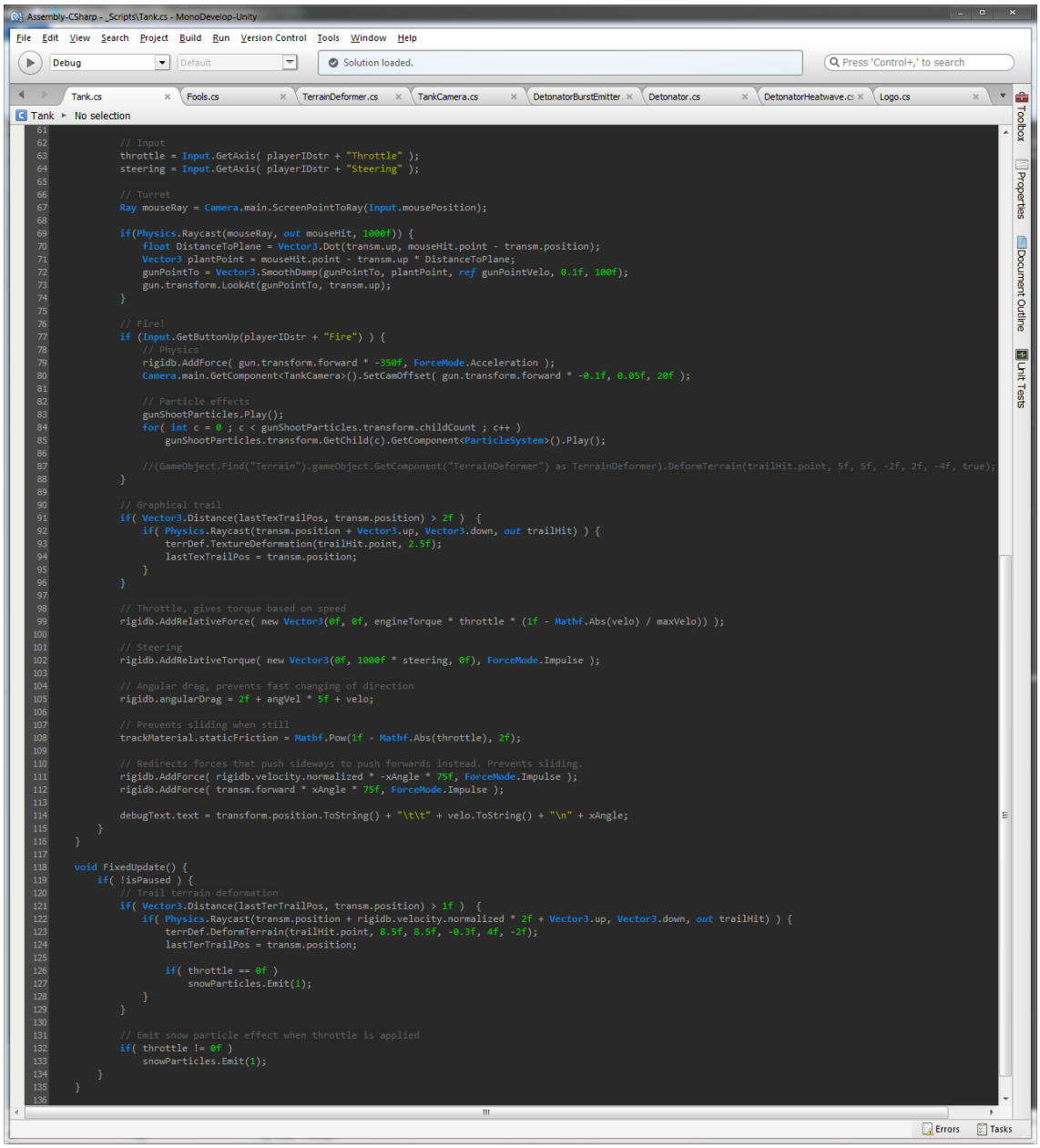
Editorin voi lisäksi muokata mieleisekseen. Lisätyökalujen tekemiseen ja laajentamiseen on kiinnitetty huomiota monipuolisen kirjaston avulla (7). Käyttäjien tekemiä lisäosia on ladattavissa ilmaiseksi tai maksua vastaan, esimerkiksi Unity3D:n sovelluskaupasta.

2.3 Ohjelmakoodi

Unity3D pyrkii tarjoamaan helppoa lähestymistä myös ohjelmointiin mahdollistamalla ohjelmakoodin kirjoituksen kolmella erilaisella kielellä (8). Selvästi harvinaisin näistä kielistä on Pythonia mukaileva Boo. Kaksi muuta vaihtoehtoa ovat C# ja JavaScriptiä mukaileva UnityScript. Tarvittaessa samaa ohjelmaa on myös mahdollista kirjoittaa eri kielillä.

Unity3D:n oma dokumentaatio on esimerkillinen kaikenkattavuudellaan ja esimerkkikoodeillaan, joskin osa koodiesimerkeistä on saatavilla ainoastaan UnityScriptinä. Lisäksi Unity3D:n suosion ansiosta internetistä on helppoa löytää ratkaisuja ja lisädokumentaatiota erinäisiin ongelmiin.

Unity3D:n mukana toimitettava ohjelmointiympäristö on Windowsissa, Linuxissa ja Macissa toimivan MonoDevelopin erikoisversio (kuva 2). Ominaisuuksiin lukeutuu modernien ohjelmointiympäristöjen ominaisuudet, kuten syntaksin väritys ja automaattinen tekstintäyttö. Ohjelmakoodin kirjoittamiseen voi luonnollisesti käyttää myös mieleistään ohjelmistoympäristöä, joskin mukana toimitettavassa MonoDevelopissa on erikoisuutena virheenkorjaustoiminto. Unity3D:n mukana toimitettavaa MonoDevelopia ei päivitetä samaa tahtia kuin tavallista MonoDevelopia.



KUVA 2. Unity3D:n mukana toimitettava MonoDevelop

2.4 Versionhallinta

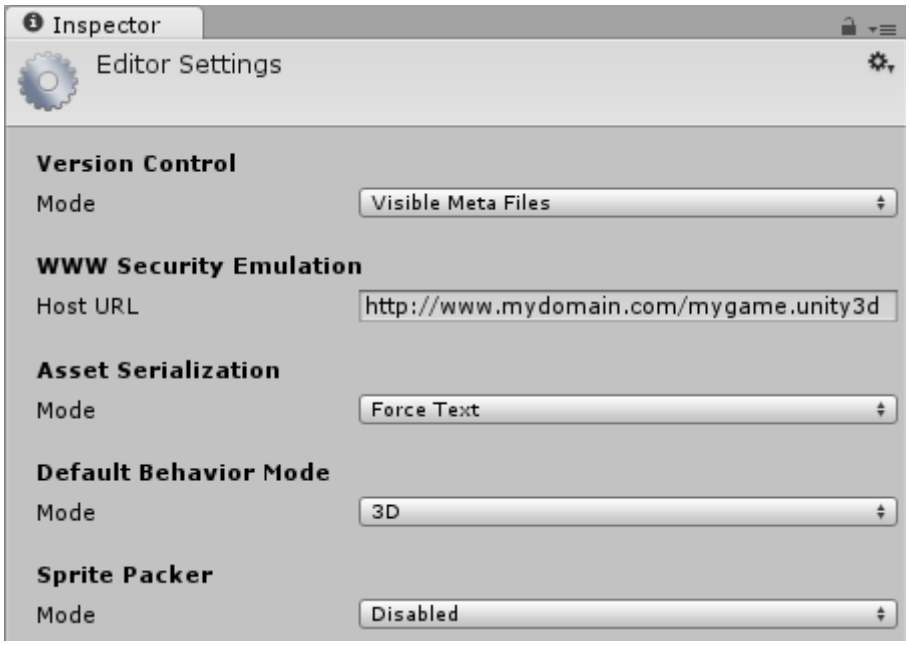
Unity3D:n ammattilaislisenssiin on mahdollista ostaa Team license -palvelu, jonka mukana tulee kehittyneempi sisäänrakennettu versionhallinta (5). Valitettavasti tätä vaihtoehtoa ei tarjota missään muodossa ilmaisversiossa.

Ilmaisversion versionhallinta oli pitkään rajoittunut pelkkiin ohjelmakoodeihin, koska Unity3D:n omia tiedostoja on pystynyt tallentamaan ainoastaan binääri-

muodossa ennen vuoden 2013 lopussa julkaistua 4.2-versiota. Uuden version myötä tiedostoja voi tallentaa myös tekstinä, jota yleisimmät versionhallintaohjelmistot ymmärtävät.

Erillisen versiohallinnan toimintaan saattamiseksi projekti tulee ensin vaihtaa pois binäärimuodosta tekstimuotoon ja siirtää se käyttämään metatiedostoja (kuva 3). Asetus löytyy Edit-pudotusvalikon alavalikosta Project Settings kohdasta Editor. Tämän jälkeen projektin kansiosta löytyvät Assets ja ProjectSettings -kansiot asetetaan versionhallinnan alaisiksi. Muut kansiot, mukaan lukien juurihakemisto, tulee jättää pois, koska Unity Editor pystyy luomaan nämä tiedostot uudestaan. (9.)

Unity Editor vahtii ja ylläpitää projektin tiedostorakennetta ollessaan päällä, eli se tulee sulkea ennen kuin versionhallintaa voi käyttää. Käytön jälkeen kannattaa ajaa Assets-pudotusvalikosta löytyvä Reimport All -komento, joka lataa projektin tiedostot kiintolevyltä uudestaan editoriin.



KUVA 3. Erillisen versiohallinnan käyttöönotto projektin Editor-asetuksista

3 PERUSOMINAISUUDET

Unity3D:n sisäänrakennetut ominaisuudet kattavat normaalit peleissä käytetyt ominaisuudet. Lisäominaisuuksia on mahdollista niin tehdä itse kuin myös ladata tai ostaa kolmansilta osapuolilta (10).

3.1 Fysiikka

Unity3D:n fysiikkamoottori perustuu Unity3D:n neljänteen versioon asti Nvidian PhysX-fysiikkamoottorin toiseen versioon, jota Unity Technologies on muokannut Unity3D:hen sopivaksi. Unity 5:n myötä PhysX päivittyy versioon 3, joka on kehittäjien internet-päiväkirjan mukaan tarkempi ja joissakin tapauksissa jopa kaksi kertaa nopeampi (11).

Objekti muuttuu fysiikkoja noudattavaksi objektiksi kun siihen lisätään Rigidbody-komponentti. Objekti alkaa vuorovaikuttamaan muiden objektien kanssa, kun siihen liitetään Collider-komponentti. Collider-komponentti määrittää objektin fysikaalisen olemuksen. Collider-komponenttina voi matemaattisesti laskettavien muotojen lisäksi käyttää myös 3D-mallia. 3D-malli fysikaalisena olemuksena on kuitenkin laskennallisesti raskas ja epätarkka, verrattuna yksinkertaisiin matemaattisiin muotoihin, joten usein kannattaa kokeilla käyttää useaa matemaattiseen kuvioon perustuvaa Collider-komponenttia 3D-mallin sijaan (12).

Lähtökohtaisesti realistisia fysiikoita tavoiteltaessa fysiikkamoottorin parametreiksi tulee antaa realistisia SI-järjestelmän arvoja painolle, koolle ja vaikuttaville voimille. Yksi mittayksikkö pelimaailmassa vastaa yhtä metriä fysiikkamoottorille. Realististen arvojen poikkeuksena on huomioitava parametrien float-tyyppi, joka tarkoittaa sitä suurempaa virhemarginaalia, mitä enemmän luvut poikkeavat toisistaan.

3.2 Hiukkastehosteet

Unity3D:n hiukkasmoottori uudistui vuonna 2012 version 3.5 mukana täysin. Hiukkastehosteiden luominen tapahtuu Unity Editorissa parametreja muokkaamalla, jolloin tehosteiden muutokset myös näkyvät tosiajassa. Hiukkaskompo-

nentti on siitä harvinainen komponentti, ettei sen kuvaajien parametreihin pysty vaikuttamaan ohjelmakoodin avulla.

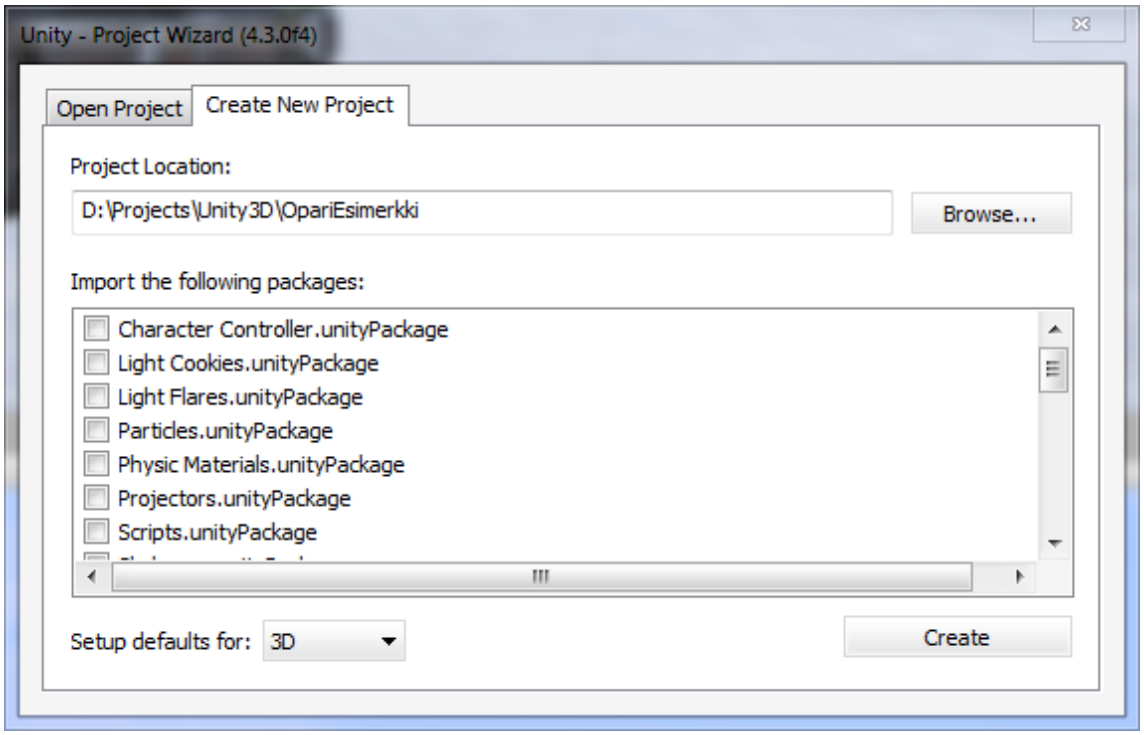
Jokainen hiukkaskomponetti luo aina tehostetta piirrettäessä uuden DrawCall-kutsun, mutta samasta komponentista luodut tehosteet piirtyvät samalla DrawCall-kutsulla (13). Hiukkasmoottorin aiheuttamia DrawCall-kutsuja voidaan minimoida piirtämällä samanlainen hiukkastehoste samalla hiukkaskomponentilla eri paikoissa. Tämä vaatii ParticleSystem.Simulate()-käskyn ajon ennen piirtoa, koska hiukkaskomponentin sijainti ilman sitä interpoloituu eikä tehostetta voida muutoin kutsua moneen paikkaan ruudulla tarkasti (14).

3.3 Äänet

Unity3D tukee useita ääniformaatteja, mukaan lukien MOD-musiikkia. Äänet kannattaa liittää Unity Editoriin pakkaamattomassa muodossa ja jättää pakkaaminen Unity Editorille. Unity Editor pakkaa äänet mobiilialustoille MPEG-formaatissa ja muille alustoille Ogg Vorbis -formaatissa (15).

4 KÄYTTÖÖNOTTO

Uusi projekti aloitetaan valitsemalla File-pudotusvalikosta New Project, joka käynnistää uuden projektin luontia varten Project Wizard -ikkunan (kuva 3). Project Wizard -ikkunassa asetetaan projektin kotikansio ja lisäpaketteja, kuten esimerkiksi valmiita hiukkas- tai äänitehosteita. Paketin lisäys projektiin siirtää paketin lisämateriaalin projektin kotikansioon ja voi viedä paljon tilaa sekä hidastaa projektin luontia. Lisäpaketteja voi tuoda myös projektin luonnin jälkeen.



KUVA 4. Projektin luonti Project Wizard -ikkunassa

4.1 Tiedostojen lisäys ja poisto projektista

Unity Editor tarkkailee projektin kotikansiota jatkuvasti, ja tiedostot yleensä ilmestyvät Project-välilehteen heti niiden lisäyksen jälkeen. Jos näin ei jostain syystä tapahdu, voi kotikansion tarkastaa painamalla hiiren oikeaa nappia Project-välilehdellä ja valitsemalla Refresh-toiminnon avautuvasta valikosta.

Kotikansion tarkkailusta huolimatta erityisesti tiedostojen poisto kannattaa tehdä Project-välilehteä käyttäen, koska Unity Editor luo projektin tiedostoille väliai-

kaistiedostoja ja ylläpitää tiedostojen yhteyksiä niitä käyttäviin komponentteihin. Virheen sattuessa edellä mainitulla Refresh-toiminnolla Unity Editor tarkastaa kansiorakenteesta myös poistetut tiedostot.

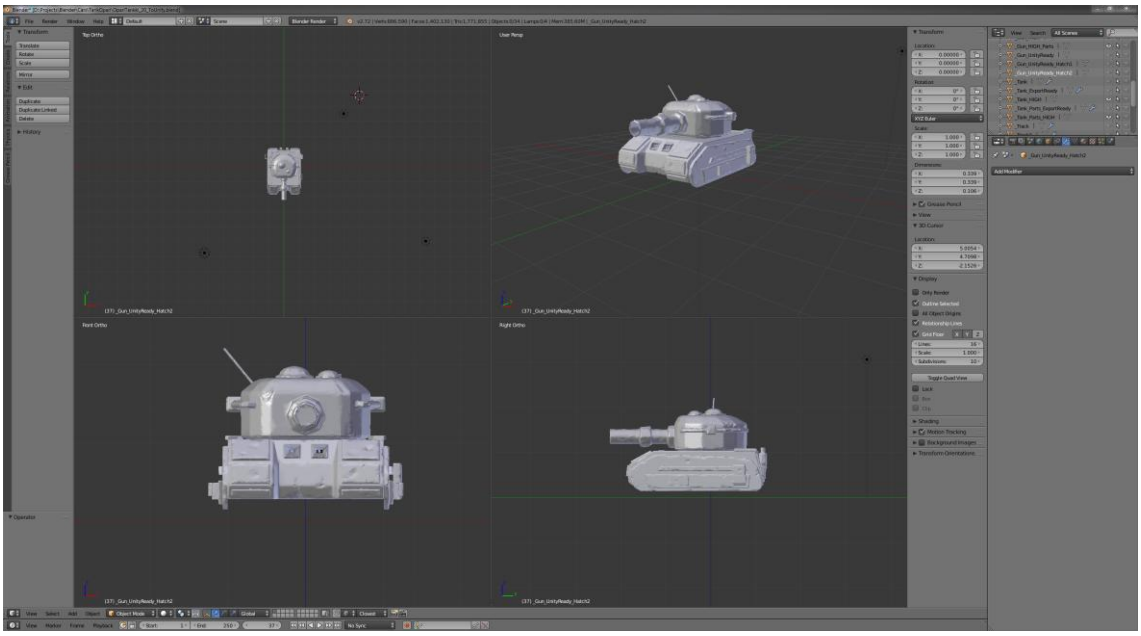
4.1.1 3D-mallien tuonti Blenderistä

Voittoa tavoittelemattoman Blender Foundationin ylläpitämästä avoimeen lähdekoodiin perustuvasta Blenderistä on tullut yksi suosituimmista ilmaisista 3D-mallinnusohjelmistoista. Unity Technologies on ottanut tämän huomioon tukemalla Blenderin omaa tiedostoformaattia, joskin pinnan alla Unity Editor ainoastaan kääntää Blenderin omat tiedostot FBX-tiedostoiksi (16). Käännöksestä johtuen Blender-tiedostojen ensi kertaa Unity Editorissa aukaiseminen voi viedä enemmän aikaa kuin vastaavan FBX-tiedoston. Yhteensopivuus on molemminpuolista ja Blenderissä on sisäänrakennettuna omat asetukset mallien vientiin Unity Editorille suoraan FBX-muotoisena.

Merkittävin yhteensopivuusongelma on koordinaatistossa, joka on Unity3D:ssä vasenkätinen ja Blenderissä oikeakätinen. Käytännössä se tarkoittaa, että Z- ja Y-akselit vaihtavat paikkaa, ja muunnoksesta johtuen mallit saattavat joskus avautua Unity Editorissa takaperin.

Blender suosii monikulmioiden käyttöä pintoina, mutta Unity3D:n kannalta mallit kannattaa muuttaa kolmioiksi, koska tällä tavoin niin näytönohjain kuin Unity3D:kin käsittelee mallien pintoja (17). Monikulmioina kolmioiden etupuoli voi osoittaa väärään suuntaan ja jäädä piirtämättä, tai malli voi näyttää oudolta, koska kolmiot ovat muutoin väärässä järjestyksessä.

Blenderiä voi mallinnuksen lisäksi käyttää Normal Map -tekstuurien luontiin. Normal Map on tekstuuri, joka simuloi valontaittoa mallin pinnalla ja luo illuusion suuremmasta tarkkuudesta kuin mitä malli oikeasti on (18). Nomal Mapia varten luodaan matalatarkkuuksisesta mallista korkeatarkkuuksinen malli, joka Normal Map -tekstuuria luodessa projisoidaan matalatarkkuuksisen mallin pintaan (kuva 5).

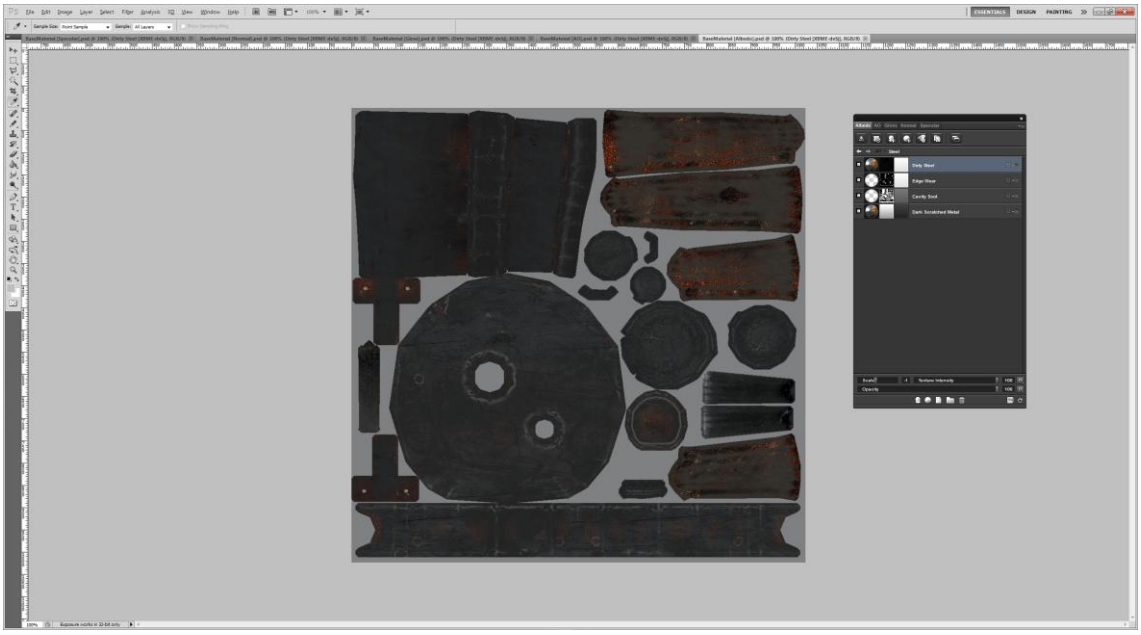


KUVA 5. Projektissa käytetyn tankin korkeatarkkuuksinen malli Blenderissä

4.1.2 Kuvien tuonti Photoshopista

Kuvankäsittelyohjelmia on monenlaisia vaihtoehtoja kaupallisista ilmaisiin, mutta silti Adobe Systemsin vuonna 1988 julkaisema kaupallinen Photoshop on pysynyt tunnetuimpana vaihtoehtona. Unity Editor tukee Photoshopin omaa tiedostoformaattia, mikä helpottaa käyttämään kaikkia Photoshopin tarjoamia ominaisuuksia ja nopeuttaa työnkulkua vähentämällä tiedostojen muuntoa eri tiedostoformaatteihin. Kaupallisuudesta johtuen tässä työssä on käytetty vuonna 2010 julkaistua CS5-versiota.

Photoshopin lisäosakirjastosta ruotsalaisen Quixelin DDO-lisäosa on huomionarvoinen aputyökalu tekstuureiden luontiin (kuva 6). DDO:ssa on sisäänrakennetut asetukset Unity3D:tä varten, jotka varmistavat, että tekstuurit ovat yhteensopivat. Hyvän yhteensopivuuden tueksi DDO:ssa on jo asetukset tulevan Unity3D:n 5.0-version uusia ominaisuuksiakin varten.

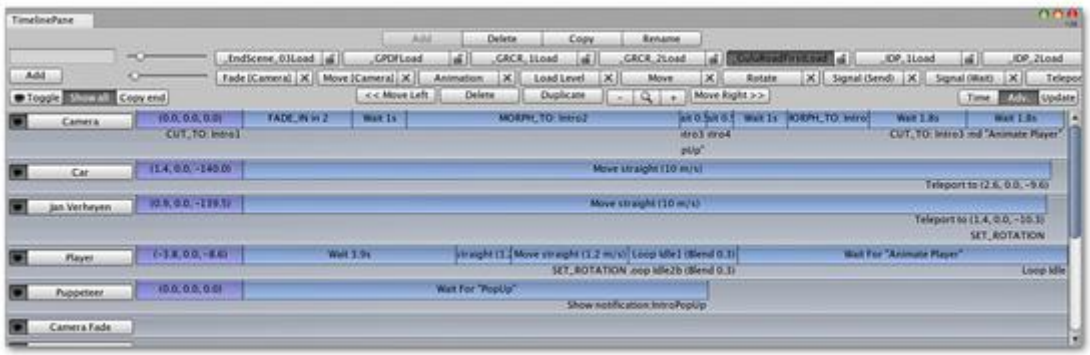


KUVA 6. Tekstuurien luonti Photoshopissa DDO:n kanssa

4.2 Unity Editorin käyttöliittymän muokkaus

Unityn editorin käyttöliittymässä on riittävät ominaisuudet peruskäyttöön. Objekteja voi lisätä ja liikutella 3D-näkymässä, ja tämän lisäksi Inspector-työkalun kautta pystyy luomaan toimintoja ja muokattavia muuttujia. Yksinkertaisin tapa tuoda inspectoriin muokattavia muuttujia on määrittää muuttuja julkiseksi ohjelmakoodissa (19).

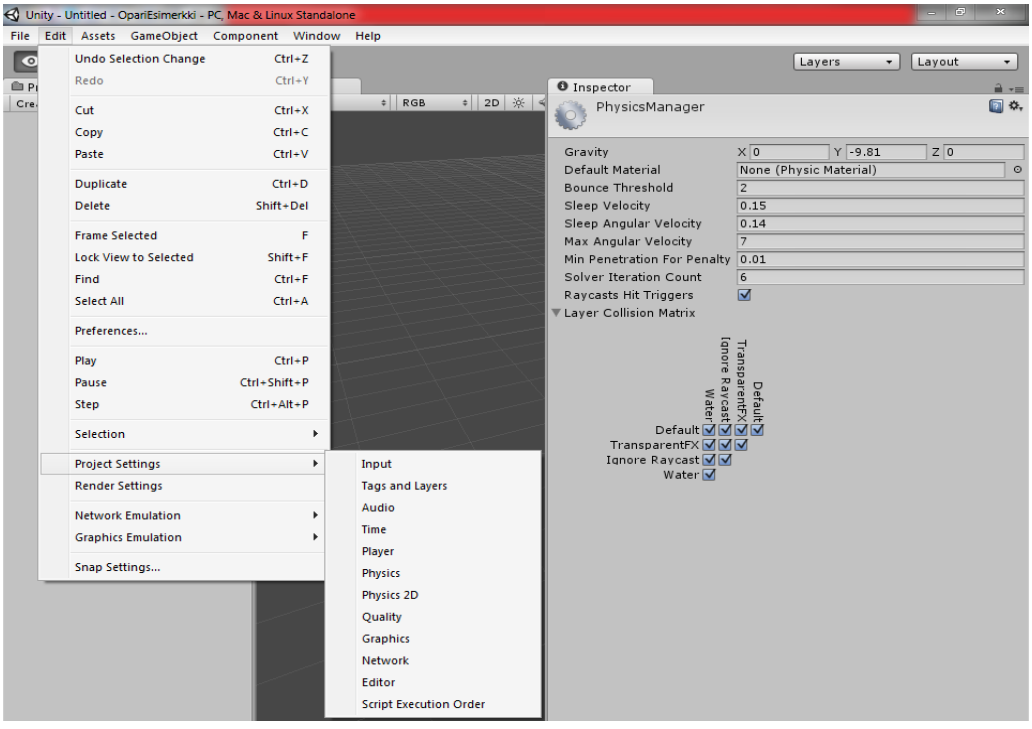
Lisäominaisuuksia tai käyttöä helpottavia ominaisuuksia varten voi niin 3D-näkymää kuin Inspector-työkaluakin muokata ja käyttää hyväksi ohjelmakoodin avulla (kuva 7). Käyttöliittymää muokkaavat ohjelmakoodit laitetaan projektin kansioon nimeltä Editor ja niiden tulee ottaa käyttöön UnityEditor-nimiavaruus (20).



KUVA 7. Serious Games Interactiven käyttöliittymää muokkaava välianimaatio-editori

4.3 Projektin ja scenen asetukset

Yläpalkin Edit-pudotusvalikon alta löytyvät tärkeät, mutta usein ylenkatsotut projektin asetukset (21). Niiden kautta voidaan muokata näppäimiä, layereitä, tageja, ääniä, aikaa, pelaajaa, fysiikoita (kuva 8), grafiikoita, yhteyksiä, editoria ja ohjelmakoodien ajojärjestystä. Tämän lisäksi projektin asetusten alta löytyvät scenen eli ympäristön asetukset, kuten epäsuora valaistus, taivas ja sumu.



KUVA 8. Valikko projektin asetuksille ja fysiikka-asetukset

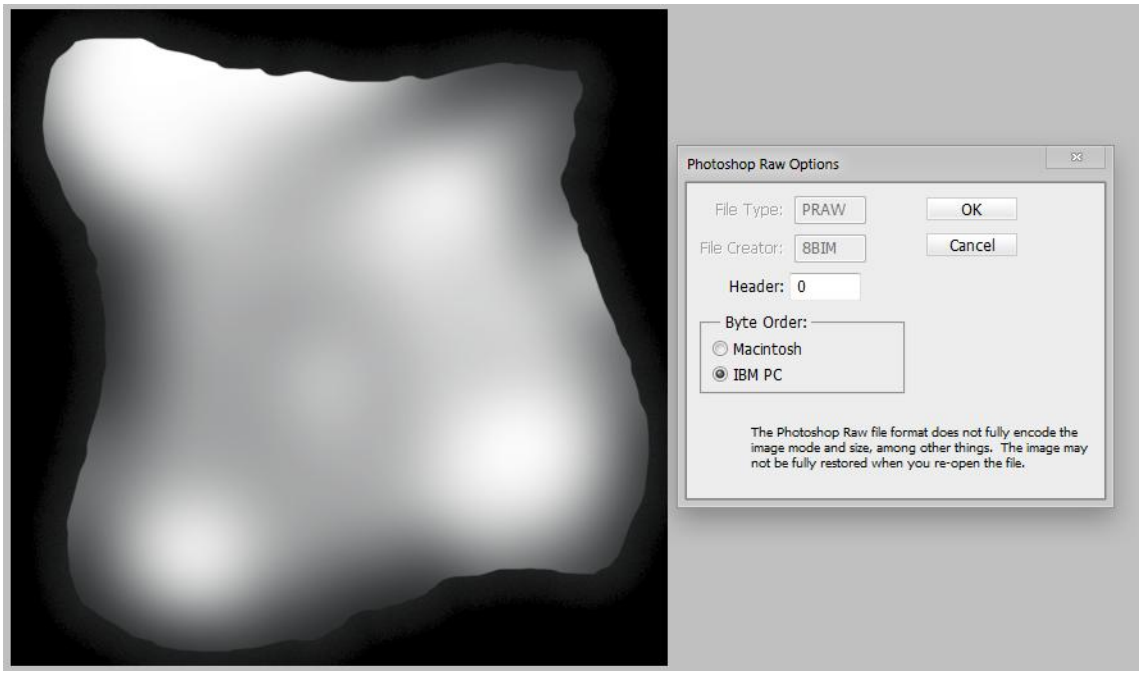
5 ESIMERKKIPELI

Esimerkkipeli keskittyi Unity3D:n oman Terrain-komponentin käyttöön. Peliympäristö on myrskyinen jäälohkare. Jäälohkareella ajetaan tankilla ja se muokkautuu tosiaikaisesti tankin liikkuessa. Projektissa käytettiin ilmaisen Unity3D:n 4.3-versiota.

5.1 Maaston lisäys

Unity3D:n Terrain-komponentin sisältävä maasto lisätään yläpalkin GameObject-pudotusvalikon alavalikosta Create Other valitsemalla Terrain. Editorissa pystyy muokkaamaan maaston pintakuviota mieleisekseen erilaisten pensseleiden avulla. Maaston pintakuvion pystyakselin tarkkuus on aina 8-bittiä eli 256 porrastusta.

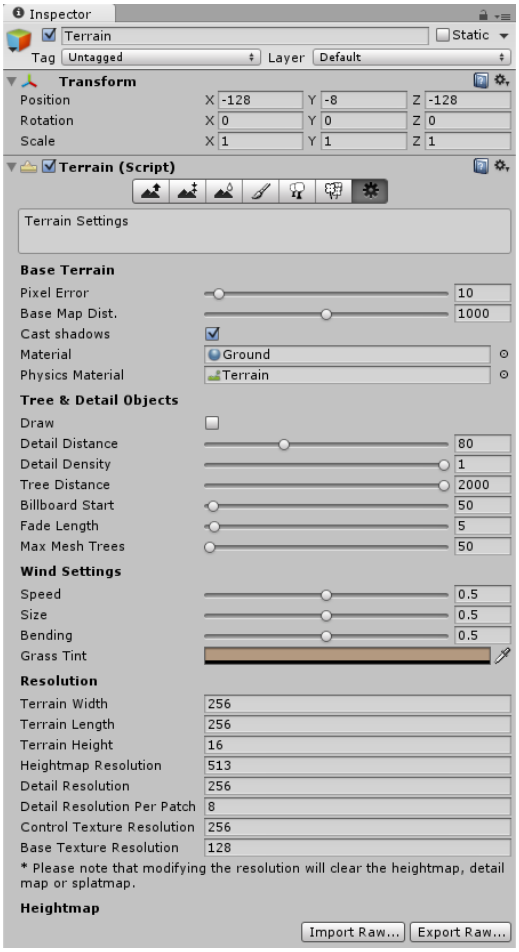
Pintakuvion pohjana voi toimia myös harmaakuva (kuva 9). Harmaakuva tallennetaan RAW-muodossa ja sen tulee vastata pintakuvion tarkkuutta. Jotta harmaakuva olisi varmasti oikeanlainen, voi Unity Editorin luoman tasaisen pintakuvion tallentaa ja avata Photoshopissa muokattavaksi (22).



KUVA 9. Pintakuvion harmaakuva ja sen tallennus Photoshopissa

Pintakuvion muokkauksen lisäksi maastoa voi myös värittää erilaisilla tekstuureilla. Maastoon pystyy myös lisäämään puita ja kasvillisuutta samojen pensseleiden avulla, joilla pintakuvion muokkaaminenkin on mahdollista. Peli sijoittuu jäälohkareelle, joten Terrain-komponentin ruohoa tai puita ei käytetä ollenkaan.

Pelin maaston leveydeksi ja pituudeksi valittiin esimerkissä 256 yksikköä (kuva 10). Tarkkuudeksi valittiin 512 yksikköä. Koska pelissä yksi yksikkö vastaa yhtä metriä, tarkoittaa se, että maaston jokaista metriä kohden on kaksi muokkauskohtaa eli toisen sanoen maastoa voi muokata puolen metrin tarkkuudella.



KUVA 10. Projektissa käytetyn maaston asetukset

Pelin maaston korkeudeksi valittiin 16 yksikköä. Koska maaston tarkkuus on aina 256 yksikköä, tarkoittaa se, että maastoa voi muokata pystyakselilla 16/256 metrin tarkkuudella eli 6,25 senttimetrin askelin. Luku voi vaikuttaa pie-

neltä, mutta koska tankin on tarkoitus pakata maastoa kasaan liikkuessaan, tarvitaan pieniä ja pehmeitä askelluksia.

5.2 Maaston tosiaikainen muokkaus

Maaston muokkaukseen pohjana toimii Devin Reimerin TerrainDeformer-luokka (1). Luokassa on perustoiminnot maaston pintakuvion ja tekstuurin muokkaukseen. Toiminnoltaan se on komponentti, joka lisätään objektille, jossa on Terrain-komponentti.

Lisäsin luokkaan käskyn, joka muokkaa maastoa hienovaraisemmin ja mahdollistaa muokkauksen syvyyden rajoittamisen suhteessa muokkaamattomaan maastoon (kuva 11). Tekstuureja muokkaavaa komentoa päädyin ainoastaan muuttamaan niin, että reunoja kohti se pehmentää muutosta tekstuurien välillä.

```
public void DeformTerrain(Vector3 pos, float craterWidth, float craterHeight, float craterDepth, float craterSteepness, float clampDepth = 0f) {
    Vector3 terrainPos = GetRelativeTerrainPositionFromPos(pos,terr,hmWidth,hmHeight);
    int heightMapCraterWidth = (int)(craterWidth * (hmWidth / terr.terrainData.size.x));
    int heightMapCraterLength = (int)(craterHeight * (hmHeight / terr.terrainData.size.z));
    int heightMapStartPosX = (int)(terrainPos.x - (heightMapCraterWidth * 0.5f));
    int heightMapStartPosZ = (int)(terrainPos.z - (heightMapCraterLength * 0.5f));

    float[,] heights = terr.terrainData.GetHeights(heightMapStartPosX, heightMapStartPosZ, heightMapCraterWidth, heightMapCraterLength);

    float   xSteepness = 1f / heightMapCraterWidth,
            zSteepness = 1f / heightMapCraterLength;

    craterDepth *= hmDepthScale;
    clampDepth *= hmDepthScale;

    for( int x = 0 ; x < heightMapCraterWidth ; x++ ) {
        for( int z = 0 ; z < heightMapCraterLength ; z++ ) {
            float   xPos = (float) (x+1) / (heightMapCraterWidth + 1),
                    zPos = (float) (z+1) / (heightMapCraterLength + 1);
            float thisDepth = Mathf.Pow( Fools.SinX01(xPos) * Fools.SinX01(zPos), craterSteepness);

            if( !debug ) {
                if( clampDepth != 0f )
                    thisDepth = Fools.MinAbs(thisDepth, Fools.Div(Mathf.Clamp(heights[z, x] -
                                                        (heightMapBackup[heightMapStartPosZ + z,heightMapStartPosX + x] + clampDepth), 0f, hmDepthScale),
                                                        Mathf.Abs(craterDepth * thisDepth)));

                heights[z, x] = Mathf.Clamp01( heights[z, x] + thisDepth * craterDepth);
            } else
                thisDepth = 1f;
        }
    }

    terr.terrainData.SetHeights(heightMapStartPosX, heightMapStartPosZ, heights);
}
```

KUVA 11. Maastoa muokkaava käsky

Varsinaiset käskyt eivät ole laskennallisesti raskaita, mutta maastoa muokatessa niin 3D-malli kuin sitä vastaava Collider-komponentti joudutaan luomaan fysiikkamoottorille jokaisen muokkauksen jälkeen uusiksi, joka on laskennallisesti raskasta (24). Tästä johtuen pelin jäälohkare ei ole kovin iso ja sen tarkkuus ei riitä täydellisen sulavaan muokkaukseen. Vaikka maaston pintakuviota voidaan muokata 6,25 sentin tarkkuudella, ilmenee muokkauksessa myös tätäkin suu-

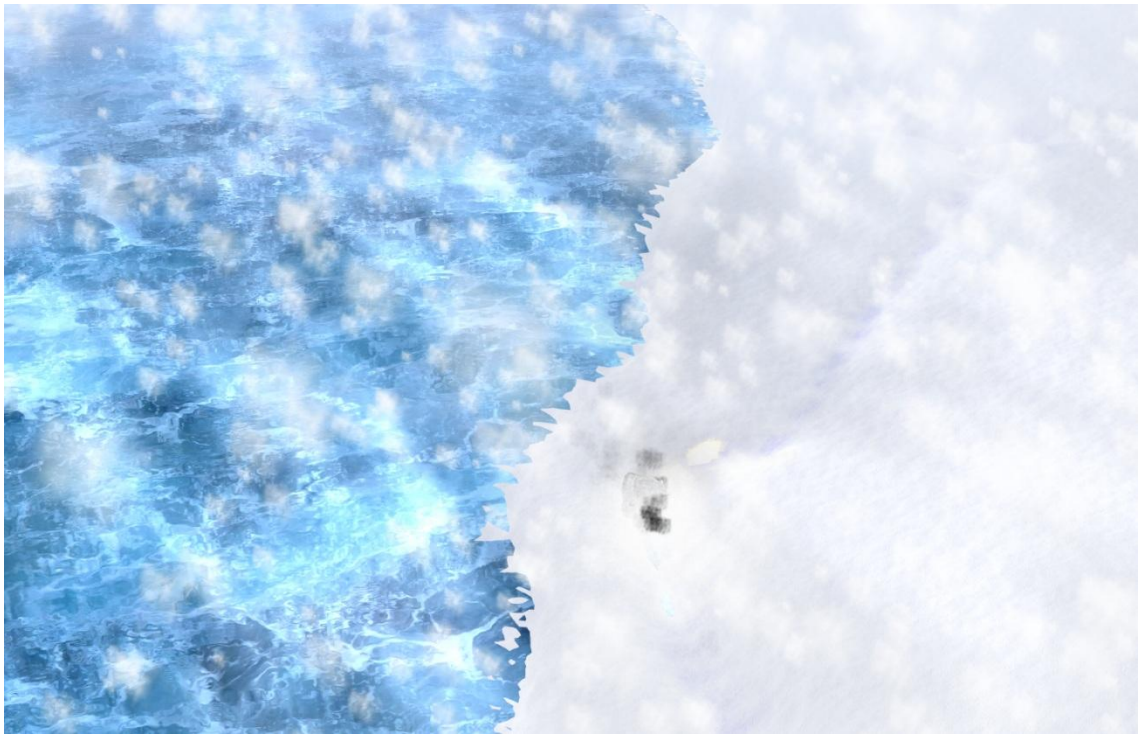
rempaa kulmikkuutta, koska maaston jatkuvaa muokkausta ei voida suorittaa tarvittavaa määrää sekunnin aikana.

5.3 Vesi

Unity3D:n mukana toimitetaan kaksi valmista vesiobjektia. Toinen näistä on realistisempi mutta vaatii ammattilaislisenssin, koska siinä valon taittumisen simuloimista varten käytetään tekstuurin piirtämistä (25). Toinen vesiobjekti on ilmaisversiossakin toimiva yksinkertainen objekti, jossa on vain liikkuva tekstuuri. Ilmaisversiossa toimiva vesiobjekti sopii periaatteessa vain mobiilipelien ulkoasuun.

Ongelma on aika yleinen, koska esimerkiksi Unity3D:n keskustelupalstalla on useita erilaisia toteutuksia vesiobjektille. Päädyin käyttämään RiverWater-komponenttia, koska siitä löytyy myös valon taittumisen simulointi (26). RiverWaterissa valon taittumisen edellyttämä tekstuuriin piirto on toteutettu hitaammalla ReadPixels-komennolla. ReadPixels-komento joutuu kierrättämään tietoa prosessorin kautta, kun taas ammattilaislisenssillä toimiva suoraan tekstuurin piirtäminen toimii täysin näytönohjaimen sisällä (27).

RiverWater otetaan käyttöön asettamalla RiverWaterin materiaali vesiobjektiin ja lisäämällä RiverWater-komponentti kameraan (kuva 12). Komponentti lisätään kameraan valon taittumisen simuloinnin vuoksi. Päädyin kuitenkin kytkemään valon taittumisen simuloinnin pois päältä, koska se laski ruudunpäivitysnopeutta 160 kuvaa pienemmäksi, noin 60 ruudun nopeuteen.



KUVA 12. RiverWaterin avulla toteutettu jäävesi

RiverWaterin mukana toimitetaan kaksi vesitekstuuria ja toista näistä muokkaamalla sain omaa silmääni miellyttävän jääveden. RiverWater käyttää pintatekstuurin lisäksi kahta muuta Normal Map -tekstuuria luomaan tehosteen liikkuvasta vesimassasta. RiverWaterin tekstuurit ovat ainoat grafiikat, joita en ole tehnyt itse.

5.4 Tankki

Tankki koostuu fyysisesti neljästä irtonaisesta kokonaisuudesta: rungosta, tykistä, tykkitornin luukuista ja teloista (kuva 13). Suunnittelin telojen animaation aluksi tapahtuvan tekstuuria liikuttamalla, mutta koska telat eivät koskaan näy merkittävästi, liitin telojen tekstuurit rungon tekstuureihin ja jätin ne staattisiksi. Telat on viritetty pomppimaan tankin mukana Configurable Joint -komponentin mukana. Configurable Joint -komponentilla voidaan luoda saranoita ja jousia (28). Tankin luukut ovat lisäksi avattavissa, mutta niitä ei käytetä.

KUVA 13. Tankki

Tankin tukevasti maassa pitäminen osoittautui erääksi esimerkkipelin haastavimmista osista, koska tankilla oli taipumus jatkuvasti kääntyä ympäri. Ratkaisu ongelmaan oli lopulta niinkin yksinkertainen kuin painovoiman kaksinkertaistaminen projektin asetuksista.

Tankin rungon tai telojen materiaalit on määritetty kitkattomiksi pinnoiksi, etteivät ne jumittuisi maastoon eli lumeen kiinni. Lisäksi jotta tankki pysyisi paikallaan pysähdyttäessä, tankin ohjelmakoodi muuttaa telojen materiaalin staattista kitkakerrointa pelaajan antaman kaasun mukaan. Tämän lisäksi tankin ohjelma-

koodi siirtää sivuttaissuuntaan kohdistuvia voimia kulkusuuntaan meneviksi, jottei se liu'u maastossa liikaa.

5.4.1 Tankin tykki

Toiminnallisesti tankin tykki seuraa hiirtä ja pyörii paikallisen y-akselinsa ympärillä. Toteutus ei ajatuksena ole monimutkainen, mutta ongelma tulee siitä, ettei objekteja voi pyörittää vapaasti kolmiulotteisessa tilassa asettamalla ainoastaan halutun akselin ympäri tapahtuva pyöritys. Vaadittujen laskutoimitusten takia kahden akselin ympäri tapahtuva pyöritys on niin monimutkainen, että esimerkiksi Unity Technologiesin omassa autokurssissa eturenkaiden kääntämistä varten eturenkaat kapseloidaan yhden ylimääräisen GameObjectin sisään (29).

Pelkkä paikallisen y-akselin ympäri tapahtuva kääntö vaatii ensiksi pistetulon laskemisen tankin pystyakselista ja vektorista, joka osoittaa tankista hiirtä kohti. Sen jälkeen otetaan vektori, joka osoittaa hiireen tankin pystyakselista, joka on edellä lasketun pistetulon pituinen. Tämän jälkeen tykin voi asettaa osoittamaan edellä saatua vektoria kohti LookAt-komennolla tankin pystyakselia käyttäen. Käytin tykin liikkeiden pehmentämistä varten myös SmoothDamp-komentoa (kuva 14).

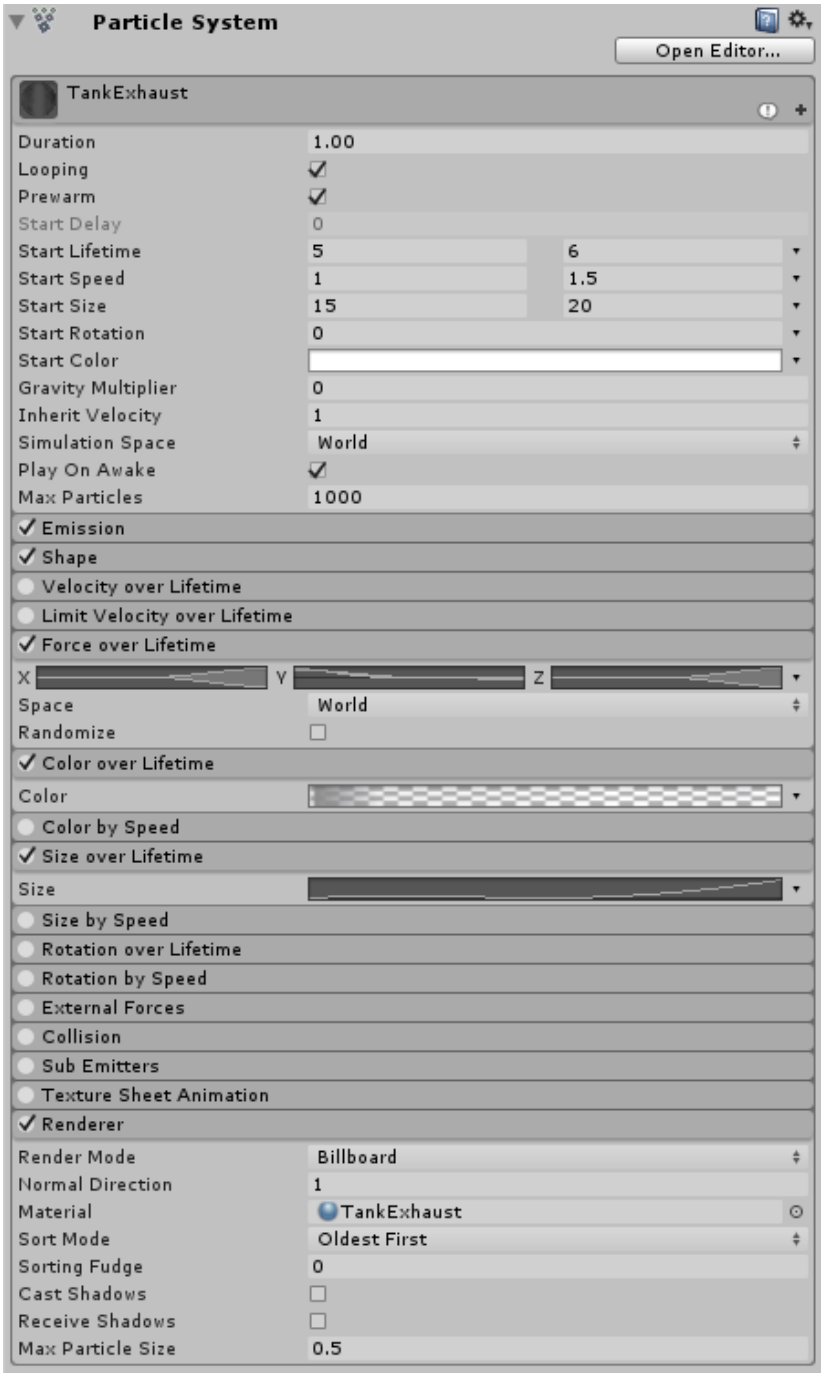
```
// Turret
Ray mouseRay = Camera.main.ScreenPointToRay(Input.mousePosition);

if(Physics.Raycast(mouseRay, out mouseHit, 1000f)) {
    float distanceToPlane = Vector3.Dot(transm.up, mouseHit.point - transm.position);
    Vector3 plantPoint = mouseHit.point - transm.up * distanceToPlane;
    gunPointTo = Vector3.SmoothDamp(gunPointTo, plantPoint, ref gunPointVelo, 0.1f, 100f);
    gun.transform.LookAt(gunPointTo, transm.up);
}
```

KUVA 14. Tykkiä pyörittävä ohjelmakoodi

5.4.2 Tankin tehosteet

Tankissa on kolme hiukkastehostetta. Ensimmäinen näistä on pakoputkista jatkuvasti tuleva tumma savu. Savu muodostuu jatkuvasti ilmestyvistä tummista pilvistä, jotka force-muuttujan avulla suuntaavat aluksi ylös ja sitten leviävät joka suuntaan. Joka suuntaan levitessä pilvien koko kasvaa size-muuttujan määri-

tysten perusteella ja ne muuttuvat näkymättömiksi color-muuttujan avulla (kuva 15).



KUVA 15. Hiukkaskomponenttien editori

Toinen hiukkastehoste on lumipöly tankin ajaessa eteenpäin. Se on liki staattinen ja ainoastaan katoaa ilmaan samalla periaatteella kuin pakoputkien savu-

kin. Merkittävin ero on siinä, että se piirretään tankin ohjelmakoodin kautta pelaajan painaessa kaasua.

Kolmas hiukkastehoste tulee tykin ampuessa. Se koostuu kolmesta tehosteesta, joista ensimmäinen sylkee tulta eteenpäin, toinen luo tulipallon tykin piipun kohdalle ja kolmas tekee pölypilven tankin ympärille. Tykin ampuessa kamera myös heilahtaa jousimaisesti tykin vastaiseen suuntaan.

5.5 Kuvatehosteet ja kamera

Kuvatehosteet ovat tärkeitä yhtenäistämään graafista ilmettä. Aluksi kokeilin RiverWater-komponentin tekijän IndieEffects-kirjastoa, joka tuo Unity3D:n ammattilaislisenssin kuvatehosteita ilmaisversioon käyttäen samaa ReadPixels-menetelmää kuin RiverWater (30). Halutun kuvatehosteen komponentti laitetaan kameraan ja se on tämän jälkeen toimintavalmis. Valitettavasti IndieEffects osoittautui jopa hitaammaksi kuin RiverWater, laskien ruudunpäivitystä jopa 190 ruutua, noin 30 ruutuun sekunnissa.

Vaihtoehtoisena menetelmänä kokeilin tehdä koko ruudun peittävän valkean tekstuurin, jonka läpinäkyvyys suurenee ruudun keskustan kohdalla. Kameran ohjelmakoodi ohjaa tämän tekstuurin materiaalin läpinäkyvyyttä sykkien 25 ja 75 prosentin välillä. Lisäksi tein kameraan hiukkastehosteen, joka luo erikokoisia vaaleita ruudun ohi leijuvia palloja. Näin luotiin illuusio pyryttävästä lumimyrkystä yhdessä lumipyryn tekstuurin kanssa (kuva 16). Vaikka ohi leijuvia palloja on tuhansia, kokonaisuutena toteutuksen pois ja päällä oloa ei huomaa ruudunpäivityksestä.

KUVA 16. Tykki ampumassa lymypyryssä

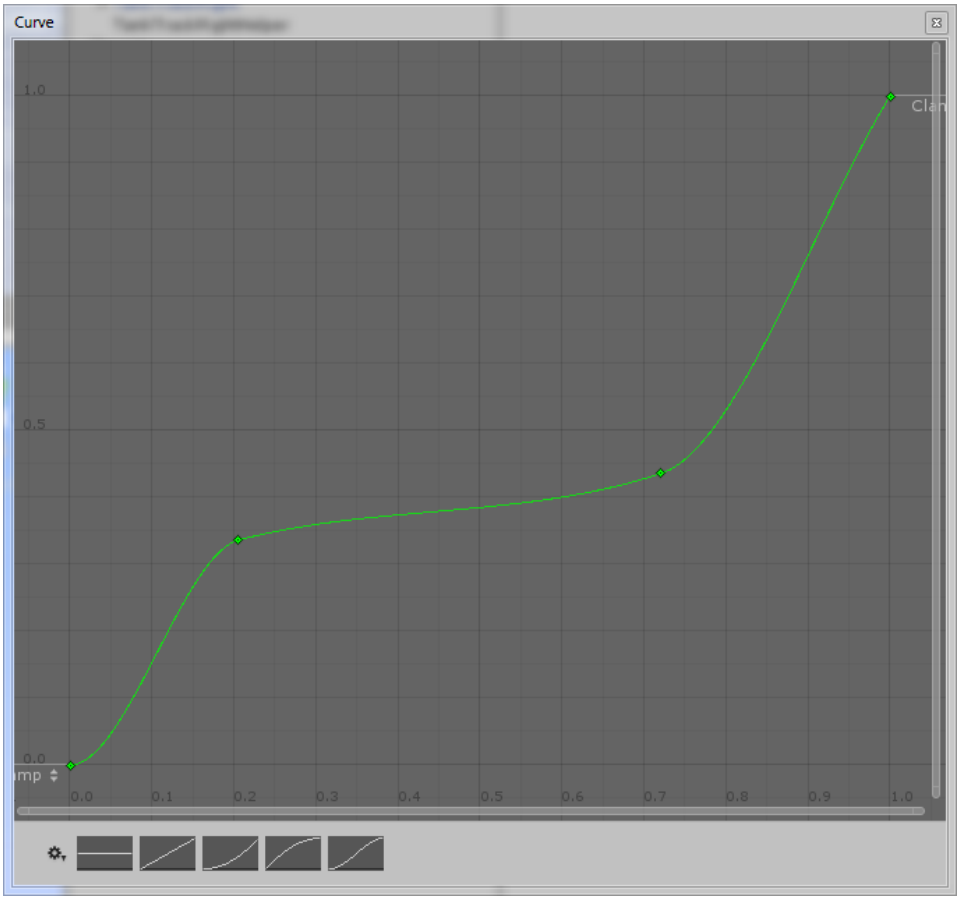
Tämän lisäksi toin DirectionalLight-objektin luomaan auringonvaloa ja lisäsin sille pilvisen Cookie-tekstuurin. Cookie-tekstuuri luo sapluunan valolle, jonka avulla kuvaan tulee enemmän variaatiota (31). Pilvet liikkuvat DirectionalLight-objektin mukana kameran ohjelmakoodin ohjaamana. Pilvien liikkuessa takaisin aloituspisteeseensä lumipyryn tekstuuri sykähtää täysin valkoisena.

5.6 Alkuteksti

Halusin saada alkutekstin ilmestymään vauhdilla ruudun vasemmasta reunasta, hidastamaan vauhtia hetkeksi ruudulle tullessaan ja sitten kiitämään pois ruudulta (kuva 17). Normaalisti vastaavaa liikettä varten tehdään matemaattinen funktio.

KUVA 17. Alkuteksti kokonaisuudessaan

Unity3D tarjoaa matemaattisten funktioiden korvaamista varten AnimationCurve-objektin (32). Lisäämällä alkutekstiä ohjaavaan ohjelmakoodiin julkinen AnimationCurve-objekti, voi funktion kuvaajan tehdä graafisesti editorin avulla (kuva 18). Kuvaajan arvo saadaan ohjelmakoodissa AnimationCurve.Evaluate-komennolla.



KUVA 18. Alkutekstin x-akselin liikkeen kuvaaja Unity Editorissa

Alkutekstien alussa kamera on sijoitettu 50 metriä normaalin korkeuden yläpuolelle, josta kamera liikkuu hitaasti alaspäin, normaaliin paikkaansa. Alkutekstiä kiertää myös PointLight-objekti valotehosteena. Alkutekstin poistuttua ruudulta se poistetaan käytöstä GameObject.SetActive-komennolla.

5.7 Yhteenveto

Unity3D:n yleisen toiminnan kanssa en kokenut vastoinkäymisiä ja sain projektin ympäristön rakennettua nopeasti. Suurimmaksi ongelmaksi yllättäen muodostui tankin tykkitornin pyöritys. Odotin toteutuksen olevan yksinkertainen Unity3D:n kirjastojen avulla, mutta se vaatikin useita laskutoimituksia.

En kokenut ilmaisversion rajoituksia merkittäviksi. Niin veden kuin lumimyrskyn toteutus onnistui vaihtoehtoisilla menetelmillä. Vesitasona käytin kolmannen osapuolen RiverWateria ja lumimyrskyyn hiukkastehosteita. Vaihtoehtoisten menetelmien tutkimiseen ja toteuttamiseen kuitenkin todennäköisesti kului ylimääräistä aikaa.

Maaston reaaliaikainen muokkaus osoittautui raskaammaksi kuin aluksi ajattelin. Muokkausta karhentamalla sain sen kuitenkin toimivaksi ja mahdollisesti lisätyöllä sen saisi vieläkin huomaamattomammaksi. Suorituskyvyn tutkimiseen tarkoitetutusta ammattilaisversion Profiler-työkalusta voisi olla apua.

6 LOPPUSANAT

Työssä esiteltiin Unity3D:n perusominaisuudet ja käyttöä pelikehityksessä. Tämän lisäksi käytiin läpi yhteensopivia työkaluja, joita myös käytettiin esimerkkiprojektissa. Unity3D on osoittautunut kattavaksi työkaluksi pelikehitykseen ja editorin monipuolisuus pienentää merkittävästi tarvetta työkalujen tekoon.

Ammattilaislisenssin erot ilmaisversioon nähden koskevat lähinnä asioita, jotka rajoittavat suurempia projekteja. Tärkein ero pienempien projektien kohdalla on ollut sisäisen versiohallinnan puute, mutta nykyisessä versioissa on mahdollista käyttää ulkoista versionhallintaa. Esimerkkiprojektissa ilmenneet Unity3D:n ilmaisversion grafiikkatehosteiden rajoitukset onnistuin kiertämään vaihtoehtoisilla ratkaisuilla.

Esimerkkiprojektissa puolet ajasta meni grafiikoiden tekoon ja puolet kehitykseen. Kehitys koostui pääosin asioiden selvittämisestä. Asioiden selvittämistä varten internetissä on useita artikkeleita Unity3D:lle. Aikajakaumassa on erityistä se, etten ole käyttänyt lainkaan aikaa minkäänlaisten editoreiden tekoon.

Maaston tosiaikainen muokkaus käyttämällä Unity3D:n omaa Terrain-komponenttia on raskasta. Lisäksi Terrain-komponentin 8-bittisen pintakuvion pystyakselin tarkkuus on matala, ja koska pintakuvio on kaksiulotteinen, luolastojen toteuttaminen ei ole mahdollista. Jatkokehitystä ajatellen luopuisin Unity3D:n omasta Terrain-komponentista ja tutkisin uuden, tarpeita vastaavan komponentin kehitystä. Unity Technologies avasi keskustelupalstallaan kesällä 2014 kyselyn Terrain-komponentin jatkokehityksestä. Kyselyn mukaan uuden version on epämääräisesti sanottu vain olevan tulossa (33).

LÄHTEET

1. Company Facts. 2015. Unity Technologies. Saatavissa: http://unity3D.com/public-relations. Hakupäivä 19.2.2015.

2. Brodkin, Jon 2014. How Unity3D Became a Game-Development Beast. Saatavissa: http://news.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/. Hakupäivä 27.2.2015.

3. Licenses Comparison. 2014. Unity Technologies. Saatavissa: http://unity3D.com/unity/licenses. Hakupäivä 3.2.2015.

4. Profiler. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/Profiler.html. Hakupäivä 19.2.2015.

5. Store. 2015. Unity Technologies. Saatavissa: https://store.unity3D.com/. Hakupäivä 19.2.2015.

6. Interface Overview. 2014. Unity Technologies. Saatavissa: http://unity3D.com/learn/tutorials/modules/beginner/editor/interface-overview. Hakupäivä 19.2.2015.

7. Scripting API. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/ScriptReference/. Hakupäivä 19.2.2015.

8. Creating and Using Scripts. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/CreatingAndUsingScripts.html. Hakupäivä 19.2.2015.

9. Using External Version Control Systems With Unity. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/ExternalVersionControlSystemSupport.html. Hakupäivä 19.2.2015.

10. Asset Store. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/AssetStore.html. Hakupäivä 19.2.2015.

11. High-performance physics in Unity 5. 2014. Unity Technologies. Saatavissa: http://blogs.unity3D.com/2014/07/08/high-performance-physics-in-unity-5/. Hakupäivä 5.2.2015.

12. Colliders. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/CollidersOverview.html. Hakupäivä 19.2.2015.

13. Draw Call Batching. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/DrawCallBatching.html. Hakupäivä 19.2.2015.

14. Moving a ParticleSystem. 2012. Banderous. Saatavissa: http://forum.unity3D.com/threads/moving-a-particlesystem-why-does-unity-interpolate-its-position.134283/. Hakupäivä 19.2.2015.

15. Audio Files. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/AudioFiles.html. Hakupäivä 19.2.2015.

16. Importing Objects From Blender. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/HOWTO-ImportObjectBlender.html. Hakupäivä 19.2.2015.

17. FBX export guide. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/HOWTO-exportFBX.html. Hakupäivä 19.2.2015.

18. Hajioannou, Yanni 2013. What Is a Normal Map. Saatavissa: http://gamedevelopment.tutsplus.com/articles/gamedev-glossary-what-is-a-normal-map--gamedev-3893. Hakupäivä 19.2.2015.

19. Kanne, Virgil 2008. Fun With Exposed Variables In Unity. Saatavissa: http://www.vergilkanne.com/design08/fun-with-exposed-variables-in-unity/. Hakupäivä 19.2.2015.

20. Special Folder Names. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/SpecialFolders.html. Hakupäivä 19.2.2015.

21. Settings Managers. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/comp-ManagerGroup.html. Hakupäivä 19.2.2015.

22. Terrain Settings. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/terrain-OtherSettings.html. Hakupäivä 19.2.2015.

23. Reimer, Devin 2010. Real-time Terrain Deformation in Unity3D. Saatavissa: http://blog.almostlogical.com/2010/06/10/real-time-terrain-deformation-in-unity3D/. Hakupäivä 6.1.2015.

24. Deformable terrain test. 2013. Moment Studio. Saatavissa: http://www.scrapsgame.com/deformable-terrain-test-dont-get-hopes-much/. Hakupäivä 19.2.2015.

25. How Do I Use Water. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/HOWTO-Water.html. Hakupäivä 19.2.2015.

26. River Water for Unity Free. 2014. FuzzyQuills. Saatavissa: http://forum.unity3D.com/threads/riverwater-the-free-epic-water-solution-for-unity-free-users.235860/. Hakupäivä 18.1.2015.

27. Jordan, Alex 2014. Recreating Unity Pro Features in Unity Free. Saatavissa: http://www.gamasutra.com/blogs/AlexJordan/20140224/211490/Recreating_Unity_Pro_Features_in_Unity_Free.php. Hakupäivä 19.2.2015.

28. Configurable Joint. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/class-ConfigurableJoint.html. Hakupäivä 19.2.2015.

29. Car Tutorial. 2012. Unity Technologies. http://u3D.as/content/unity-technologies/car-tutorial/1qU. Hakupäivä 1.2.2015.

30. IndieEffects. 2014. FuzzyQuills. Saatavissa: http://forum.unity3D.com/threads/indieeffects-bringing-almost-aaa-quality-post-process-fx-to-unity-indie.198568/. Hakupäivä 19.2.2015.

31. How Do I Make A Spotlight Cookie. 2014. Unity Technologies. Saatavissa: http://docs.unity3D.com/Manual/HOWTO-LightCookie.html. Hakupäivä 19.2.2015. Hakupäivä 19.2.2015.

32. Laine, Leo 2014. Animation Curves and Unity. Saatavissa: http://tomodomo.org/animation-curves-and-unity/. Hakupäivä 19.2.2015.

33. New Terrain System. 2014. Unity Technologies. Saatavissa: http://forum.unity3D.com/threads/official-new-terrain-system.255232/. Hakupäivä 11.2.201